Claims

1. A method of analyzing program execution within an operating system of a multithreaded environment, comprising:

accumulating diagnostic data pertaining to a thread accessing a resource, the execution of a thread being predicated upon the thread's access to the resource;

storing the diagnostic data within a data structure at a location in the data structure correlated to the resource.

2. The method according to claim 1, wherein the diagnostic data includes data selected from a group consisting of: a time measurement, program code executed by the thread, an invocation stack, pointer data and some combination, thereof.

3. The method according to claim 1, wherein the data structure comprises a hash bucket.

4. The method according to claim 1, further comprising determining the resource.

5. The method according to claim 4, wherein determining the resource includes reading contents of a task dispatcher.

6. The method according to claim 1, further comprising storing information identifying the resource.

7. The method according to claim 1, further comprising matching an identifier corresponding to the resource to a correlative identifier corresponding to the data structure.

8. The method according to claim 7, further comprising reassigning the identifier to a second resource.

9. The method according to claim 7, further comprising assigning the correlative identifier to the data structure.

10. The method according to claim 1, further comprising detecting a locking occurrence.

11. The method according to claim 10, further comprising calculating a time increment corresponding to a duration that the thread remains locked.

12. The method according to claim 11, further comprising storing the time increment within the data structure.

13. The method according to claim 10, further comprising recording the time corresponding to the locking occurrence.

14. The method according to claim 1, further comprising detecting a removal of the lock.

15. The method according to claim 14, further comprising recording a time instance corresponding to the removal of the lock.

16. The method according to claim 10, further comprising recording program data relating to code executed by the thread prior to the locking occurrence.

17. The method according to claim 16, further comprising retrieving the program data from an invocation stack.

18. The method according to claim 1, further comprising displaying the diagnostic data.

19. A method of analyzing program execution within a computer system having a plurality of threads accessing a plurality of resources, comprising:

calculating a time increment reflective of a duration a thread of the plurality of threads waits for access to a resource of the plurality of resources, the execution of the thread being predicated upon the thread's access to the resource; and

storing the time increment within a bucket of a plurality of buckets comprising a hash array, each bucket being correlated to the resource.

20. The method according to claim 19, further comprising reallocating the plurality of resources to the plurality of buckets to group the diagnostic data with a different scheme.

21. An apparatus comprising:

at least one processor configured to execute a plurality of threads;

a memory; and

program code resident in the memory and configured to execute on the at least one processor, the program code configured to accumulate diagnostic data pertaining to a thread accessing a resource, the execution of a thread being predicated upon the thread's access to the resource, and to store the diagnostic data within a data structure at a location in the data structure correlated to the resource.

22. The apparatus according to claim 21, wherein the diagnostic data includes data selected from a group consisting of: a time measurement, program code executed by the thread, an invocation stack, pointer data and some combination, thereof.

23. The apparatus according to claim 21, wherein the lock of memory comprises a hash bucket.

24. The apparatus according to claim 21, wherein the program code initiates a determination of the resource.

25. The apparatus according to claim 21, wherein the program code initiates storing information identifying the resource.

26. The apparatus according to claim 21, further comprising matching an identifier corresponding to the resource to a correlative identifier corresponding to the data structure.

27. The apparatus according to claim 26, wherein the program code initiates reassigning the identifier to a second resource.

28. The apparatus according to claim 26, wherein the program code initiates assigning the correlative identifier to the data structure.

29. The apparatus according to claim 21, wherein the program code initiates a detection of a locking occurrence.

30. The apparatus according to claim 21, wherein the program code initiates a calculation of a time increment corresponding to a duration that the thread remains locked.

31. The apparatus according to claim 30, wherein the program code initiates storing the time increment within the data structure.

32. The apparatus according to claim 21, wherein the program code initiates recording a time corresponding to a locking occurrence.

33. The apparatus according to claim 21, wherein the program code initiates detecting a removal of the lock.

34. The apparatus according to claim 33, wherein the program code initiates recording a time instance corresponding to the removal of the lock.

35. The apparatus according to claim 29, wherein the program code initiates recording program data relating to code executed by the thread prior to a locking occurrence.

36. The apparatus according to claim 35, wherein the program code initiates retrieval of the program data from an invocation stack.

37. The apparatus according to claim 21, wherein the program code initiates a display of the diagnostic data.

38. A program product, comprising:

program code for analyzing program execution within an operating system of a multithreaded environment, wherein the program code is

configured to accumulate diagnostic data pertaining to a thread accessing a resource, the execution of a thread being predicated upon the thread's access to the resource, and to store the diagnostic data within a block of the memory correlated to the resource; and

a signal bearing medium bearing the program code.

39. The program product of claim 38, wherein the signal bearing medium includes at least one of a recordable medium and a transmission-type medium.